Assignment 1 - Trade-off between Overfitting and Underfitting

Due Date: October 9, 11:30 pm

Late submission

If you submit the assignment after the deadline, the following penalty is applied:

- ◇ 10% penalty if the submission is before October 14, 11:30 pm (if the mark before applying the penalty is 78 out of 100, after applying the penalty it is 78 - 7.8 = 70.2 out of 100);
- ◇ 50% penalty if the submission is after October 14, 11:30 pm and before Dec. 8, 11:30 pm.

Description:

In this assignment you are required to perform an experiment similar to the experiment in Section 1.1 of PRML [1]. You will have to compare several regression models to illustrate the trade-off between overfitting and underfitting. You will use data generated synthetically. The prediction is to be performed based on only one feature, denoted by x. The target t satisfies the following relation

$$t = \underbrace{sin(4\pi x)}_{f_{true}(x)} + \epsilon \tag{1}$$

where ϵ is random noise with a Gaussian distribution with 0 mean and variance 0.09.

Construct a training set consisting of only 10 examples $\{(x^{(1)}, t^{(1)}), \cdots, (x^{(10)}, t^{(10)})\}$, where $x^{(1)}, \cdots, x^{(10)}$ are uniformly spaced in the interval $[0, 1]$, with $x^{(1)} = 0$, $x^{(10)} = 1$, and $t^{(1)}, \cdots, t^{(10)}$ are generated using relation (1). Construct a validation set consisting of 100 examples with features generated randomly using a uniform distribution over the interval $[0, 1]$ (or just uniformly spaced) and targets generated randomly according to relation (1). You may use the code provided on the next page. **When generating the random data use the number formed of the last 4 digits of your student ID, as seed for the pseudo number generator.**

You have to train ten regression models of increasing capacity (corresponding to M from 0 to 9) and will record and compare their training and validation errors. For each M, $0 \leq M \leq 9$, the predictor has the form

$$f_M(x) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M.$$

Note that, when $M = 0$, the predictor is just a constant function. For each M you have to train the predictor using least squares and record the training and validation errors. Then plot the prediction $f_M(x)$ function and the curve $f_{true}(x)$ versus $x \in [0, 1]$. Additionally, include in the figure all the points in the training set and in the validation

set with their true target values. Use different colours for the training examples, the validation examples, the prediction function and the true function.

Additionally, for $M = 9$ you have to train the model with regularization in order to control overfitting. Here you have to try several values for λ until you find a value λ_1 that eliminates the overfitting. You also have to find a value λ_2 for which underfitting occurs. For each of λ_1 and λ_2 plot the prediction $f_M(x)$ function and the curve $f_{true}(x)$ against x, and all the points in the training set and in the validation set with their true targets.

You have to write a report to present your results and their discussion. The report should contain all the plots mentioned above and a plot of the training and validation errors versus M, for all twelve predictor functions that you build. Also include in this plot the average error between the targets and the true function $f_{true}(x)$ for the examples in the validation set. The report should be clear and concise.

Besides the report, you have to submit your numpy code. The code has to be modular. Write a function for each of the main tasks. Also, write a function for each task that is executed multiple times (e.g, to compute the average error). The code should include instructive comments.

Code for data generation:

You may use the following code for generating the training and the validation sets:

```
import numpy as np
X_train = np.linspace(0.,1.,10) # training set
X_valid = np.linspace(0.,1.,100) # validation set
np.random.seed(...)
t_valid = np.sin(4*np.pi*X_valid) + 0.3 * np.random.randn(100)
t_train = np.sin(4*np.pi*X_train) + 0.3 * np.random.randn(10)
```

Code for feature standardization:

When using training a linear model with regularization, the features have to be standardized first. You may use the code given below for feature standardization.

For $M = 9$, the linear regression model is applied on new feature vectors with 9 features. Specifically, from each x we obtain the new feature vector $(x, x^2, x^3, \cdots, x^9)$. The matrix XX_train used in the following code has on each row the new feature vector corresponding to one training example. Thus, its dimension is 10-by-9. Likewise, XX_valid contains on each row the feature vector for one validation example.

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
XX_train = sc.fit_transform(XX_train)
XX_valid = sc.transform(XX_valid)
```

Note that the standardization is performed based on the training data. Then the same transformations are applied to the validation data in order to be able to use the trained predictor on the validation data. In other words, the value that is subtracted from each feature and the value used for scaling in the validation data are the same ones that were used for the training data.

Submission Instructions:

- Submit the files in the Assignments Box on Avenue. Specific instructions for the format of the submitted files will follow.

References

[1] C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006 (ISBN 9780387310732), available for free download at `https://www.microsoft.com/en-us/research/people/cmbishop/prml-book/`.